A MAPPING MODEL FOR TRANSFORMING TRADITIONAL SOFTWARE DEVELOPMENT METHODS TO AGILE METHODOLOGY

Rashmi Popli[1], Anita[2] and Dr. Naresh Chauhan[3]

Department of Computer Engineering, YMCA University of Science & Technology, Faridabad Haryana, India
rashmimukhija@gmail.com
anitaarora_20@rediffmail.com
nareshchauhan19@yahoo.com

ABSTRACT

Agility is bringing in responsibility and ownership in individuals, which will eventually bring out effectiveness and efficiency in deliverables. Agile model is growing in the market at very good pace. Companies are drifting from traditional Software Development Life Cycle models to Agile Environment for the purpose of attaining quality and for the sake of saving cost and time. Nimbleness nature of Agile is helpful in frequent releases so as to satisfy the customer by providing frequent dual feedback. In Traditional models, life cycle is properly defined and also phases are elaborated by specifying needed input and output parameters. On the other hand, in Agile environment, phases are specific to methodologies of Agile - Extreme Programming etc. In this paper a common life cycle approach is proposed that is applicable for different kinds of teams. The paper aims to describe a mapping function for mapping of traditional methods to Agile method.

KEYWORDS

Agile Software Development, Pair programming, Mapping function.

1. INTRODUCTION

In the last few years Agile software development life cycle appeared as a reaction to traditional ways of developing software and acknowledges the need for an alternative to documentation driven, heavyweight software development processes. Lifecycle of any model is the time span between activities that comprises of release of first version to last version (at customer desk). Software effort needed for development follows one lifecycle. The aim of software development [4, 15] is to utilize the resources and time to its fullest but not at the cost of sacrificing quality. Lifecycle models provide a starting point for defining what will be done. There is a big difference between process and life cycle. A process is a sequence of steps for achieving any goal. A process refers to the specific steps used in a specific organization to build systems. It indicates the specific activities that must be undertaken and artifacts that must be produced. The process definitions include more detail than provided lifecycle models.

More traditional approaches, like the Waterfall model and its variances, facilitate knowledge sharing primarily through documentation. They also promote usage of role based teams and detailed plans of entire software development lifecycle. The allocation of work specifies "not only what is to be done but how it is to be done and the exact time allowed for doing it". This shifts the focus from individuals and their creative abilities to the processes themselves. These traditional approaches are often referred to as "plan-driven" or "task-based". In contrary, agile methods namely extreme programming [6, 9, 16] emphasize and value individuals and interactions over processes**.**

Dynamic business environments for software development and the need to anticipate even late requirement changes led to many different management approaches [17]. Since the establishment of the Agile Manifesto in 2001 they have been called agile methods [18]. These methods promise higher customer satisfaction, better adherence to delivery dates, better working climate for developers and so on.

1.1. Agile Manifesto

According to Agile Manifesto [5] 'individuals and interactions over processes and tools" is the foremost proverb while working in agile environment. It means processes and tools are of less value as compared with individuals and interactions. On the other side, in traditional approaches like waterfall, spiral, prototyping etc. process remains fixed. Phases are properly defined and documented so that any one can follow this static approach.

This static and dynamic approach of traditional and agile models makes this issue debatable as transition is taking place from waterfall to agile in most of the software companies. Agile software development [10] is adopted as a required support for attaining quality and quality is respected by each and every customer. To ensure this change, software industry requires many concerns to be discussed namely top level management interest, infrastructure needed, resources attitude and many more. For accommodating Agile in the software industry a mapping is required that need to be discussed so that transition can take place between two software development life cycles in proper manner.

In this paper, a mapping has been presented so that transition task can be achieved with convenience of team members and top management. In this paper, Section II summarizes the phases of different traditional approaches in comparison with Agile, Section III is about proposed mapping between Agile and traditional models, Section IV list out the benefits of this mapping and how it can be functional and Section V briefs the conclusion and future work.

2. SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

Traditional methodologies try to be predictive **-** to create a schedule at the beginning of a project and to conform to this schedule for the life of the project. Complex software systems can be built in a sequential, phase-wise manner where all of the requirements are gathered at the beginning, all of the design is completed next, and finally the master design is implemented into production quality software. This approach holds that complex systems can be built in a single pass, without going back and revisiting requirements or design ideas in light of changing business or technology conditions. Yet a common complaint is "the problem with this project is that the users keep changing their minds". In the physical world people accept that requirements need to be fixed because it's intuitively obvious to them that, because of the expensive construction phase, it's very expensive to make changes after a certain point.

2.1. Linear Sequential Model

It was first proposed in 1970 by W.W. Royce. Royce advocated iterations of waterfalls adapting the results of the precedent waterfall. In it, development flows steadily through requirements analysis, design implementation, coding, testing, integration, and maintenance [4]. This model formed the basis for the many software frameworks. In it, with every phase one deliverable is compulsory. This is basically document driven model in which proper sequence is maintained. Problem of this classic approach is mainly inflexibility which is related with change in customer mind set by seeing changing needs of time. Also, bugs keeps on propagating from one phase to another. On the other side, in agile environment requirements keep on changing as per the market need and business value.

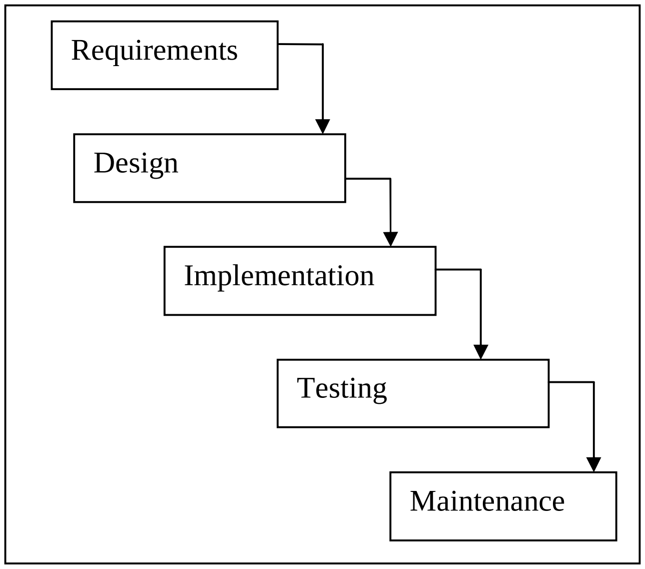

Figure 1. The Classic Waterfall Model

2.2. The Prototyping Model

The process of this model involves many small activities viz identification of basic requirements, developing of initial prototype, review and enhancing of prototype. There are two types of prototyping including close-ended or throwaway prototyping and breadboard or evolutionary prototyping. In the former case, prototype of the requirement is created that will eventually be discarded rather than becoming part of the final delivered software. The main goal of latter is to build a robust prototype in structured manner and constantly refine and rebuilt it. This acknowledges that work on those requirements that are well understood. The problem with this prototyping approach of software development is the cost and time as in the former case prototype is thrown away after reviewed by the customer. Also, presence of customer may be an issue.

In Agile way of software development, customer interactions are more important and either customer or one of the representative of customer is always present with the team members so that feedback can be received for the improvement at any time and requirements can be modified by changing trends of market. Communication is the basis for the good quality of the software.

Also, prototype here is not created rather stories are developed and demo is shown to the customers.

2.3. Iterative Incremental Model

Incremental model is an evolution of waterfall model. The product is designed, implemented, integrated and tested as a series of incremental builds. The Incremental software development model may be applicable to projects where:

- Software Requirements are well defined, but realization may be delayed.
- The basic software functionality are required early.

It generates working software quickly and early during the software life cycle. It is more flexible - less costly to change scope and requirements. It is easier to test and debug during a smaller iteration. Also, it is easier to manage risk because risky pieces are identified and handled during its iteration. In this case, problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

In agile context, builds are gradually created. Then review is done with the help of demonstrations to the customer. Also, review is possible within the existing team members or product owners.

2.4. Spiral Model

The Spiral model was first defined by Barry Boehm. It combines elements of evolutionary, incremental, and prototyping models. The term spiral refers to successive iterations outward from a central starting point. The goal of it is to identify risk and focus on it early. In theory, risk is reduced in outer spirals as the product becomes more refined. Each spiral

- starts with design goals
- ends with the client reviewing the progress thus far and future direction
- was originally prescribed to last up to 2 years

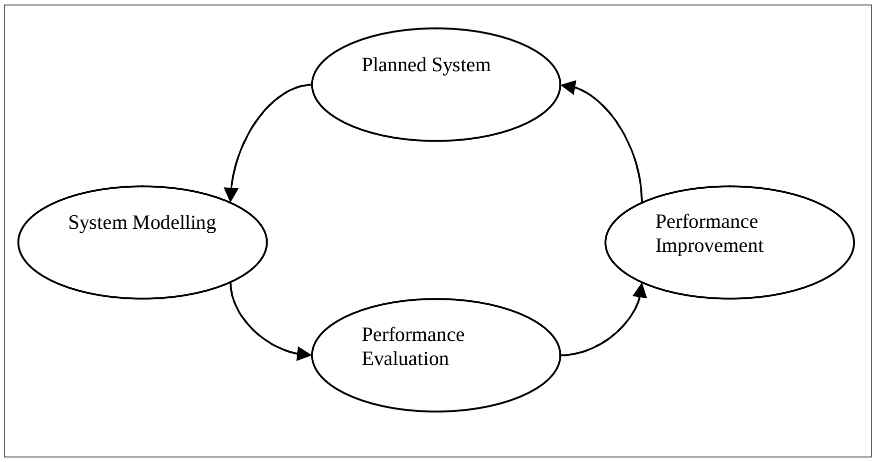

Figure 2. Spiral Model

The basic concepts of spiral model are Planned system, System Modeling, performance evaluation, performance evaluation as shown in Figure 2. Major applications of spiral model are mainly high risk projects in which requirements are not clearly understood, architecture is not clear, quality issues, and problem in the underlying technology. Problem with this model is its management cost as it is complex way of software development. Also, amount of documentation required in intermediate stages is a tough affair.

In agile, manifesto says that working software is more desirable over comprehensive documentation. Here, displaying of information is preferred as compared with keeping piles of documents.

3. Agile Software Development Life Cycle

Agile is a very recent software development methodology (or more correctly, a group of methodologies) based on the Agile manifesto. This was developed to solve some shortcoming in traditional software development methodologies. Agile methods are based on giving high priority to the customer participation early in the development cycle. It recommends testing by the customer in every phase and often as possible. Testing is done at each point when a stable version becomes available. The foundation of agile is based on starting testing from the beginning of the project and continuing throughout to the end of the project. The Agile methods, concentrate (1) more on individuals and interactions than processes and tools, (2) more on working software than comprehensive documentation, (3) value customer collaboration more than contract negotiation, and (4) focus more on responding to change than following a plan. The Scrum and Extreme programming are two of the most popular variations of agile methods.

Agile SDLC contains the six phases as shown in Figure 3: Pre project planning (first cycle), Start, Construction, Release, Production and Retirement. In Pre-project planning phase, firstly the goals of the project and market aspects are defined. It explores how the new functionality improves the organization’s presence in the market, how it will impact profit of organization, and how it will impact the people of an organization. It helps in identifying the potential stakeholders and their goals. Then in the Start phase the requirement modeling is done. In this phase active participation of stakeholders is needed to identify the initial requirement modeling or high-level, requirements for the system. Main goal of Start phase is to understand the problem and solution domain.

During Construction iterations, high-quality working software is delivered incrementally, which meets the changing needs of the user or customer. In Agile Software Development change in the requirements is allowed to meet the exact needs of the customers. The stakeholders are given complete control over the scope, budget, and schedule – they get what they want and spend as much money as required and for as long as they’re willing to do so. At the end of each development cycle or iteration there is a partial, working system to show it to the customer. Pre-production testing can be done like system integration testing. During the Release iteration phase, also known as the "end game", the system is transit into production. The goal of the Production Phase is to keep systems useful and productive even after the product has been deployed to the user community. The goal of the Retirement Phase is the removal of a system release from production, and occasionally even the complete system itself. This activity also known as system decommissioning.

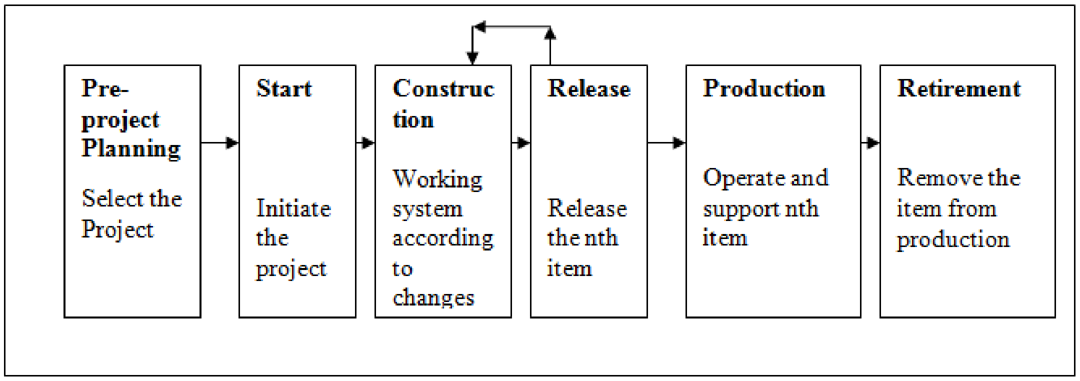

Figure 3. Agile Life Cycle

4. PROPOSED AGILE MODEL

Accepting change is the mandatory requirement for the agile teams. Without accepting change agile cannot exist in industry. Issue is how to do the transition when one of the traditional models is the heart of the company and everybody has expertise in that. One thing is for sure that if somebody is ready for accepting change then in the beginning things would seem to be complex but with the support of the organization / management, team and proper coach agile can be implemented with good success rate. In this section, a mapping model has been discussed by considering the existing model of the organization. Before concentrating on individual mapping of the different process models a general agile model is presented in which life cycle of the software product in agile context is elaborated. The main components (Refer Figure 4) of the life cycle are:

- Team Formation by good recruitment policy (TFR)
- Goal Building cycle with Quality Analyst, business Analyst and Customer. (GBC)
- Effort and Budget estimation (EBE)
- Coding & Testing activities with Communication (CTC)
- Demonstrations in Review with feedback (DRF)
- Risk evaluation and correction (REC)
- Satisfaction for all parties (SFP)

These components are the base of an agile culture. On this platform different pillars can be placed which can be helpful to achieve the more quality standards which are desired by the customer. Description of each and every component is given below:

TFR- In Agile working Environment, Team can be formed by devoting time by trainers in upgrading the technical and managerial skills, polishing the right kind of attitude and embracing the change from time to time (Refer Figure 5) and by following good recruitment policies to identify the right person. In the team, there can be experienced members as well as fresher but attitude is the biggest factor while doing recruitment.

GBC- Stories [8] are identified, evaluated and approved by customer, quality analyst and business analyst by considering the market demand and return on investment value. Evaluation is carried out by finding the competitive level of the existing products. Presence of QA along with customer helps in setting the template in mind so that at pair programming moment he or she can supply the right kind of feedback to the developer. Also he or she can design test cases before development can go in progress.

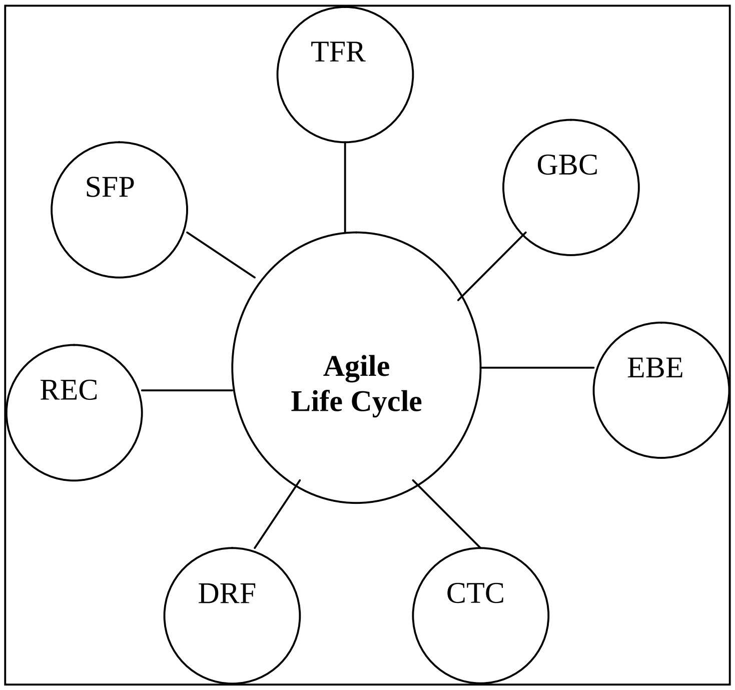

Figure 4. Proposed Agile Model

EBE- Effort and budget are estimated by considering the resource and tool requirements for each story from time to time by product owner or customer by keeping in mind the story priority. After identification of iteration or sprint stories two weeks cycle starts. In some cases, this time period of sprint can be smaller as less is always more. Estimation related with effort can be done by planning poker or any other famous technique. Estimation is possible at three scales namely iteration plan, release plan and project estimation. Estimation units are story points and ideal time. Also, velocity is very important for estimating how many stories get completed in one sprint or for finding throughput of the team.

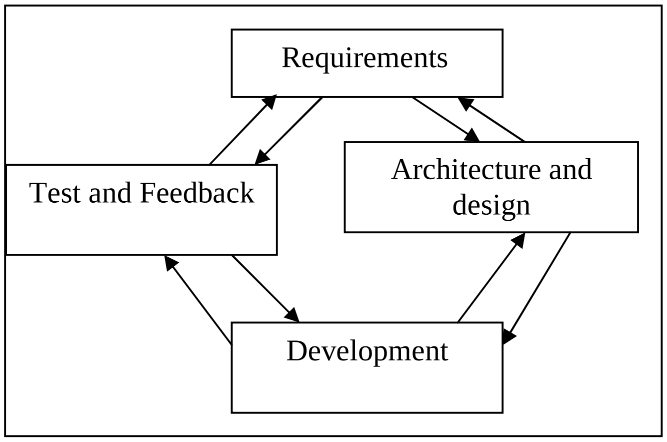

Figure 5. An Agile Team Interaction

CTC- Implementation of story starts when estimation are properly done. Pair programming (Refer Figure 6) approach [1, 2, 11] is used while doing coding and testing in which one person is the leader or executor and second person is the reviewer by working on the single terminal. This immediate feedback helps in reducing number of bugs which may otherwise keep on propagating. Remote pair programming, also known as virtual pair programming or distributed pair programming, is pair programming where the two programmers are in different locations

By pair programming knowledge get distributed, programming skills are shared and mistakes are reduced. Also, test driven development [3, 7, 12, 13, 14] (TDD) approach is used in which test cases are written before writing the code for the story.

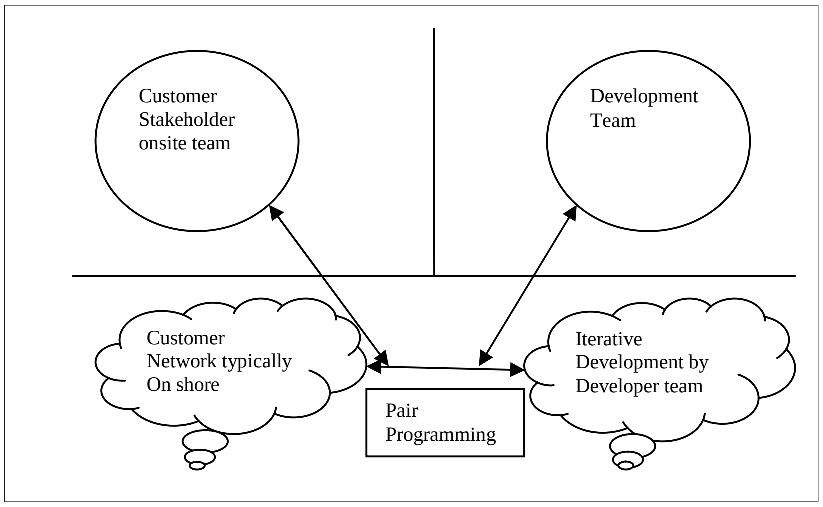
Customer Stakeholder onsite team
Development Team
Customer Network typically On shore
Pair Programming
Iterative Development by Developer team

Figure 6. Pair Programming

DRF- At the time of review, team members, management and customers sat together for the purpose of demonstrations of the software product. One of the representatives of team gives the demo for the product. Then, goal matching action is performed meaning that whether story approved has been the end product or not. In the figure 7 (shown below), two boards are there, one is for demonstration purpose and second one is for the feedback. Feedback can be given by any member including customer, management or also existing member in the team. This review meeting is informal kind of meeting.

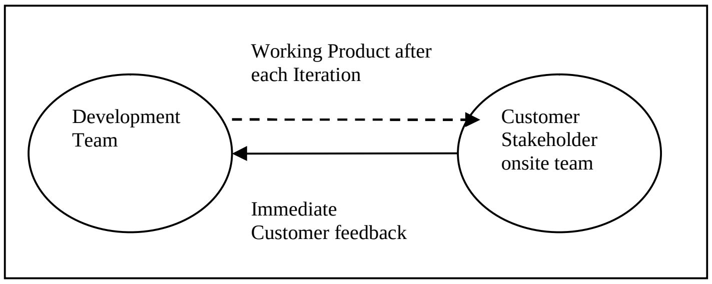
Working Product after each Iteration
Development Team
Customer Stakeholder onsite team
Immediate Customer feedback

Figure 7. Feedback System

REC- Further, in the agile cycle, risk evaluation is done for the future stories so that things can be improved or taken to the next level of quality. Actually customer is not just customer or end user rather he or she is bothered about money, quality, time and last but not the least about sustainability of the product in the market for long time. In short, he or she is worried about return on investment (ROI) [16]. So, there is a need of early detection of high risk stories so that risky outcomes are in mind before finishing the current stories. After effect of it can be significant viz chances in quality improvement to many more.

SFP- Ultimately, all parties are satisfied namely customer, team and management as product can be delivered on time by following continuous delivery, continuous feedback, continuous integration, continuous testing and continuous ROI.

Now, question of mapping arises, when there is a need to do transition from existing model to agile. Issues that may come during mapping are:

- What is the reason for the transition?
- Is management interested or customer?
- Whether team is of that much calibre or not?
- Whether infrastructure requirements are sufficient like open workspace or not?
- Whether automated tool knowledge is needed?

After resolving all these issues, work can be started for the mapping function from one model to agile. Figure 8 is for the mapping function to take place in the existing organization when transformation decision has been taken by the management. In the formula (2), mapping function MF is important. Its role is to map the large teams into small teams (T), large tasks into small stories (J), long iteration into small sprint (I), long feedback cycle into instant feedback (F), late delivery into fast small delivery (D), long meetings into daily small meetings (M), late testing into test driven testing (TG), two monitors into one terminal for pair programming (MO), estimation in lines of code into story points (E), and last but not the least project manager into no boss approach (B),co-ordination effectiveness(CE).The CE (Refer Figure 10) depends upon some implicit and explicit factors (Refer Formula (1)).

$$\text{CE}=\text{Implicit factors}+\text{Explicit factors}. \qquad (1)$$

$$\text{MF} = (T, J, I, F, D, M, TG, MO, E, B, CE). \qquad (2)$$

By using this mapping function, any of the traditional models can be converted to the agile environment. In this function, mainly ten parameters are there which are needed for the complete agile environment in an organization. Along with it at hardware ground, cubicles can be converted into open work environment, many documents can be converted into one story board, many automated tools can be converted into specific tool for specific technology/domain, and more overtime is converted into 40hrs/ week of effective work. It means in short agile is “less is more” approach which is more beneficial with less cost and time. But interactions are more through face to face communication in collocated culture and through video conference in distributed environment.

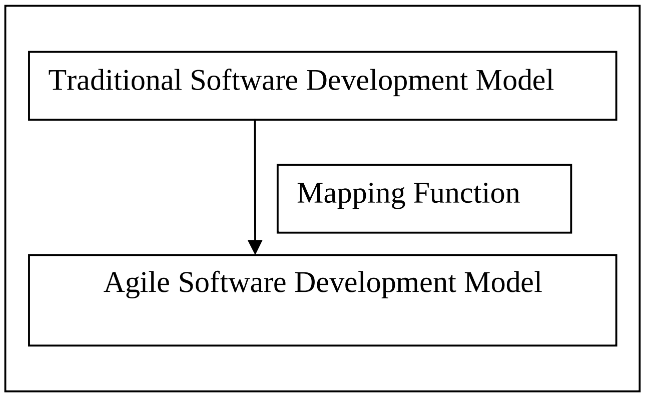
Traditional Software Development Model
Mapping Function
Agile Software Development Model

Figure 8. Mapping Function

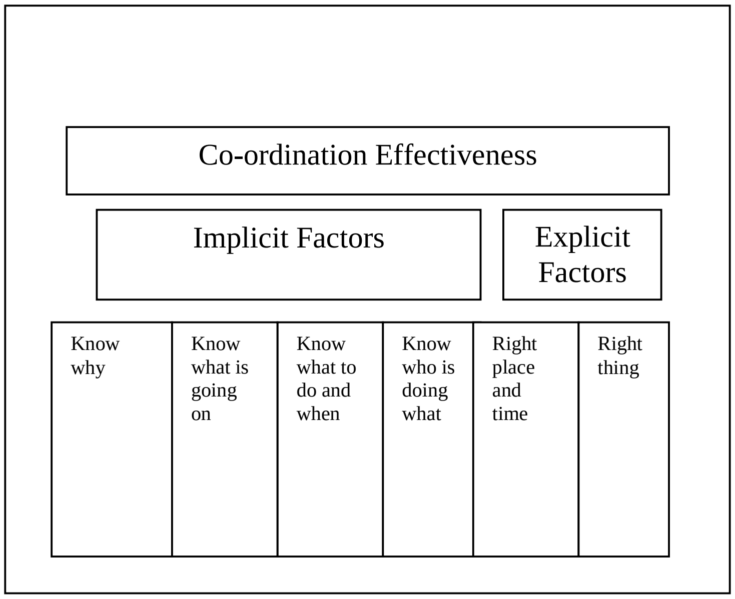

Figure 9. Factors of Coordination Effectiveness

By using this mapping function, any of the traditional models can be converted to the agile environment. In this function, mainly ten parameters are there which are needed for the complete agile environment in an organization. Along with it at hardware ground, cubicles can be converted into open work environment, many documents can be converted into one story board, many automated tools can be converted into specific tool for specific technology/domain, and more overtime is converted into 40hrs/ week of effective work. It means in short agile is “less is more” approach which is more beneficial with less cost and time. But interactions are more through face to face communication in collocated culture and through video conference in distributed environment.

5. THE BENEFITS

The proposed mapping function can be functional when parameters are identified in the existing model and transformation is done as per the mapping parameters mentioned in the previous section. If input parameters are less in existing model then transformation would not take place and left out parameters of the mapping function would be considered from the scratch in the new model i.e. Agile. Major benefits that can be reflected from this mapping function are in terms of:

- Time consumption would be less.
- Simple approach.
- Everybody would be happy (team members, client, and management).
- Old resources would not be unemployed.

6. CONCLUSION AND FUTURE WORK

In this paper, an agile model is proposed for the purpose of adopting a new process in the organization. After explaining all the components of this model, a mapping function is presented for the sake of doing transformation from one traditional model to new agile model. This mapping function is the backbone of the agile culture in the organization and success rate of any agile project can scale up by matching all the mapping parameters.

In future, on the basis of this mapping function, one tool can be designed which can identify the existing parameters in the existing traditional model and finally can do the transformation in the form of new parameters namely team size, sprint size, effort size, and how many terminals are needed and how frequently switching would take place among the team members in pair programming scenario. Also, a real project can be prepared where transition takes place easily and with less effort.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their useful suggestions that helped us to improve our work. We would also like to extend our gratitude to our university who provided us with the needed research facilities.

REFERENCES

[1] Duque, R. and Bravo, C.(2008) Analyzing work productivity and program quality in collaborative Programming, .The 3rd International Conference on Software Engineering Advances, pp.270- 276.
[2] Preston,D.,(2006) Using collaborative learning research to enhance pair programming pedagogy, ACM SIGITE Newsletter, Vol.3, No.1, pp.16-21
[3] Kent Beck,(2002), "Test Driven Development" Addison Wesley.
[4] Abran, A., Moore, J. W., Bourque, P., & Dupuis, R. (2004). Guide to the software engineering body of knowledge. Los Alamitos, CA: IEEE Computer Society.
[5] Agile Manifesto. (2001). Manifesto for agile software development. Retrieved January 1, 2009, from http://www.agilemanifesto.org
[6] Beck, K. (2001). Extreme programming: Embrace change. Upper Saddle River, NJ: Addison-Wesley.
[7] Briggs, T., & Girard, C. D. (2007). Tools and techniques for test driven learning in CS1. Journal of Computing Sciences in Colleges, 22(3), 37-43.
[8] Cohn, M. (2004). User stories applied: For agile software development. Boston, MA: Addison-Wesley.

[9] Hedin, G., Bendix, L., & Magnusson, B. (2003). Introducing software engineering by means of extreme programming. Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), Portland, Oregon, 586-593.
[10] Highsmith, J. A. (2002). Agile software development ecosystems. Boston, MA: Addison Wesley.
[11] Jacobson, N., & Schaefer, S. K. (2008). Pair programming in CS1: Overcoming objections to its adoption. SIGCSE Bulletin, 40(2), 93-96.
[12] Janzen, D. S., & Saiedian, H. (2006). Test driven learning: Intrinsic integration of testing into the CS/SE curriculum. Proceedings of the 37th ACM Technical Symposium on Computer Science Education (SIGCSE 2006), Houston, Texas, USA, 254-258.
[13] Janzen, D. S., & Saiedian, H. (2008). Test driven learning in early programming courses.Proceeding of the 39th ACM Technical Symposium on Computer Science Education (SIGCSE2008), Portland, Oregon, USA, 532-536.
[14] Kollanus, S., & Isomottonen, V. (2008). Test driven development in education: Experiences with critical viewpoints. Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education (ITICSE 2008), Madrid, Spain, 124-127.
[15] Mead, N., Carter, D., & Lutz, M. (1997). The state of software engineering education and training. IEEE Software, 14(6), 22-25.
[16] Rico, D. F. (2008). What is the ROI of agile vs. traditional methods? An analysis of extreme programming, testdriven development, pair programming, and scrum (using real options).TickIT International, 10(4), 9-18.
[17] Koch,(2005) Agile Software Development - Evaluating the methods for your Organization. Artech House Inc., Norwood, Massachusetts
[18] K. Beck et al.,(2001) Manifesto for Agile Software Development, Available at:http://agilemanifesto.org/

AUTHORS

1. RASHMI POPLI

Rashmi Popli is pursuing her Ph.D in Computer Engineering from YMCA University of Science & Technology, M.Tech(CE) from M.D University in year 2008,,B.Tech(IT) from M.D University in the year 2004.She has 9 years of experience in teaching. Presently she is working as a Assistant Professor in department of Computer Engineering in YMCA University of Science &Technology, Faridabad, Haryana, India. Her research areas include Software Engineering, Software Testing and Software Quality.

2. ANITA

Anita is pursuing her Ph.D in Computer Engineering from YMCA University of Science & Technology, M.Tech(CE) from M.D University in year 2009, B.Tech(CSE) from M.D University in the year 2004. She has 11 months of industry experience and 7 years of teaching experience. She is lifetime member of Computer Society of India and Agile Software Community of India. Her research includes Software Engineering, Software Testing and Software Quality.

DR. NARESH CHAUHAN

Naresh Chauhan received his Ph.D in Computer Engineering in 2008 from M.D University, M.Tech(IT) form GGSIT,Delhi in year 2004,B.Tech(CE) from NIT Kurukshetra in 1992.He has 20 years of experience in teaching as well as in industries like Bharat Electronics and Motorola India Pvt. Ltd.Presently he is working as a Professor in the department of Computer Engineering ,YMCA University of Science and Technology, Faridabad, Haryana, India.. His research areas include Internet Technologies, Software Engineering, Software Testing and Real Time Systems.